Adam Tecle & Matthew Robinson

Overview

This simple BitTorrent client connects with a tracker and downloads a single file. It parses the torrent file given in command line arguments and constructs an HTTP get request based on information within it. When a connection is established successfully, the tracker sends a message in response containing a current list of peers, from which the client extracts the peer with peer ID prefix '–AZ5400' – the peer dedicated to uploading our desired file. After a handshake message sent by the client, the peer and client are ready to begin exchanging messages. After information transfer is complete, the client sends a 'completed' message to the tracker and closes all communication channels.

The Tracker class encapsulates an object containing information parsed from the response of the HTTP get request to the tracker. It contains the list of peers, which are stored as an ArrayList of Peer objects, the client's peer ID, the info_hash field from the torrent file, as well as several integer fields related to the download progress of the file. It uses decoding methods in the Bencoding2 class to parse the metadata of the torrent file and construct the HTTP get request.

A tracker object, as well as the target peer contained in the ArrayList field of the tracker object, used to construct a PeerConnection object, which establishes the connections required for data transfer. The doHandshake() method of the PeerConnection class initiates the exchange of messages. After verifying the response of the handshake, our main class RUBTClient.java, repeatedly requests pieces of the file until it has received each one, then proceeds to close data input/output streams and TCP socket.

Class summaries

Tracker.java

This class encapsulates tracker object. It contains tracker info hash, client peer id, peers list, port of tracker, a TorrentInfo object, uploaded, downloaded, and 'left' fields. This class is used to communicate with tracker up until the point the client receives the peer list. It has a several helper methods that assist in intermediary

steps, such as private String randomAlphanumeric() and private String byteArraytoURLString()

RUBTClient.java

This is our main class ,which takes two command line arguments, torrent file and output file. The torrent filename argument is passed into the provided TorrentInfo constructor, after its contents are converted to byte[], A tracker object is constructed from the TorrentInfo object, the ArrayList field is populated and parsed for the desired seeder, which we send to the PeerConnection constructor to begin downloading.

Peer.java

Encapsulates Peer object, storing fields associated with a unique peer. Primarily used for creating ArrayList<Peer>.

PeerConnection.java

This class handles data transfer once the correct peer has been found within the peer list returned by the http get to tracker. The doHandshake() method is called first, which sends vital information to the peer about torrent and client. The peer responds with its peer ID once it verifies the protocol, info hash, and client peer id. The sendInterested() method is called afterwards, letting the peer know that the client is interested in downloaded a file. An unchoke message from the peer indicates that the client may begin requesting. The methods sendRequest and getPiece are called in a loop within main, until data transfer is complete.

Dependencies

Tracker.java depends on

Bencoder2.java

BencodingException.java

Peer.java

TorrentInfo.java

Peer.java has no dependencies

Adam Tecle & Matthew Robinson

RUBTClient.java depends on

All classes

PeerConnection.java depends on

Peer.java

Tracker.java

TorrentInfo.java